ИНФОРМАТИКА

УДК 004.042

В. П. Потапов, М. А. Костылев, С. Е. Попов

ПОТОКОВАЯ ОБРАБОТКА РАДАРНЫХ ДАННЫХ В РАСПРЕДЕЛЕННОЙ СРЕДЕ APACHE SPARK

Институт вычислительных технологий СО РАН, Российская Федерация,
630090, Новосибирск, пр. Академика Лаврентьева, 6

В статье описан современный подход к созданию распределенного программного комплекса на базе массово-параллельной технологии для потоковой пре- и постобработки радарных снимков. Отличительными особенностями системы являются ее способность работы в режиме реального времени с большими объемами потоковых данных, а также применение существующих алгоритмов, не предназначенных для распределенной обработки, на множестве узлов без изменения реализации последних. Проведено сравнение технологий распределенных вычислений, на основе которого делается выбор в пользу системы Apache Spark. Показано, что ее функциональность позволяет организовать автоматическую обработку поступающих радарных снимков в виде последовательности операций (workflow), которые необходимо выполнить над входными данными в зависимости от заданных ранее условий. Результаты обработки остаются доступными в системе в виде устойчивых к сбоям распределенных коллекций данных (RDD-Resilient Distributed Data), что позволяет по мере поступления космических снимков и их автоматической обработки, согласно цепочке алгоритмов, на каждом этапе получать/сохранять промежуточный результат в распределенную файловую систему HDFS. Охарактеризованы особенности имплементации конкретных задач процессинга радарных данных в рамках предложенного подхода (расчет фазы, корегистрация, формирование интерферограммы и развертка фазы методом роста регионов). Представлена блок-схема алгоритма развертки фазы с возможностью его запуска на платформах с использованием графических устройств, поддерживающих технологию NVIDIA CUDA. Представлена адаптация ее к системам с массово-параллельным исполнением заданий. Имплементация алгоритма ориентирована на вычисления для пары радарных изображений на одном вычислительном узле. Ускорение достигается за счет возможности одновременной обработки множества пар изображений, равных количеству узлов кластера. Показан пример реализаций методов работы с потоками бинарных данных (BinaryRecordsStream), осуществляющих мониторинг распределенной файловой системы HDFS на наличие поступающих радарных данных и чтение/запись их как бинарных файлов со значением фиксированного размера байт. В качестве входных параметров используются каталог и размер одной записи в байтах. В заключении приведены результаты тестирования разработанных алгоритмов на демонстрационном кластере. Показано, что при количестве узлов, равном восьми, в среднем возможно достижение 8-кратного прироста скорости работы для такого же количества пар изображений по срав-

Потапов Вадим Петрович — доктор технических наук, профессор; potapov@ict.sbras.ru
Костылев Михаил Александрович — аспирант; 5999ft@gmail.com
Попов Семен Евгеньевич — кандидат технических наук; popov@ict.sbras.ru

Potapov Vadim Petrovich — doctor of engineering sciences, professor; potapov@ict.sbras.ru
Kostylev Mikhail Alexandrovich — postgraduate student; 5999ft@gmail.com
Popov Semen Evgen'evich — PhD in engineering sciences; popov@ict.sbras.ru

 DOI: 10.21638/11701/spbu10.2017.204

нению с их последовательной обработкой на отдельном вычислительном узле. Результаты тестирования дают возможность повышения производительности представленных алгоритмов при увеличении количества узлов кластера без внесения изменений в их реализацию, что оправдывает применение распределенного подхода для решения задач пре- и постобработки радарных данных. Библиогр. 26 назв. Ил. 4. Табл. 3.

Ключевые слова: Apache Spark, Apache Hadoop, распределенные информационные системы, радарная интерферометрия, алгоритмы обработки.

V. P. Potapov, M. A. Kostylev, S. E. Popov

THE STREAMING PROCESSING OF SAR DATA IN DISTRIBUTED ENVIRONMENT WITH APACHE SPARK

Institute of Computational Technologies of the Siberian Branch
of the Russian Academy of Sciences, 6, Academician M. A. Lavrentiev pr., Novosibirsk, 630090, Russian Federation

This article presents a modern approach to creating a distributed program complex based on mass-parallel technology for pre- and postprocessing of SAR images. The unique features of the system is the ability to work in real time mode with huge amounts of streaming data and applying existing algorithms that are not used for distributed processing on multiple nodes without changing the algorithms' implementation. A comparison has been made of distributed processing technologies based on which we have selected Apache Spark. The ability to organise automatic processing of input SAR images as a sequence of operations which should be performed based on defined conditions is demonstrated. The results of processing store in the system as fault tolerant distributed collections of data (RDD-Resilient Distributed Data), which allows getting and saving the intermediate results in the distributed file system HDFS as and when new space images became available and processed by the sequence of algorithms. This article described the implementation for the specific tasks of SAR data processing based on the suggested approach is described (phase estimation, coregistration, interferogram creation and phase unwrapping with region growing method). A scheme of the phase unwrapping algorithm with the ability to use GPU and NVIDIA CUDA technology is presented. An adaptation of the algorithm for the mass-parallel systems is shown. The algorithm implementation focused on processing pair of SAR images on one node. Performance growth is achieved by simultaneous processing multiple images whose number is equal to cluster nodes count. An example of methods implementation for working with streaming binary data (BinaryRecordStream) which perform monitoring of new SAR data in distributed file system HDFS and readingthis data as binary files with fixed bytes size is shown. A directory and size of one record are used as the input parameters. The results of testing developed algorithms on demonstration cluster is presented. A possibility of getting up to eight times better processing speed using eight nodes in a cluster for the same images count in comparison with sequential processing on one node is shown. Results of testing provide the ability to improve the performance of presented algorithms without any changes in implementation and this in turn justifies the utility of applying distributed approach for SAR data processing. Refs 26. Figs 4. Tables 3.

Keywords: Apache Spark, Apache Hadoop, distributed information systems, sar interfometry, processing algorithms.

Введение. Данные дистанционного зондирования Земли получили широкое распространение практически во всех отраслях как науки, так и промышленности и используются для решения самых разнообразных задач, число которых постоянно растет. Не исключением стали и радарные данные, получаемые с космических аппаратов радиолокационной съемки.

Основным аппаратом анализа радарных данных является метод радарной дифференциальной интерферометрии (DInSAR), который занимается построением цифровых моделей рельефа (ЦМР) на основе расчета разности фаз нескольких радарных снимков одной территории [1, 2].

В то же время пре- и постобработка радарных данных обусловлена повышенными требованиями к вычислительным ресурсам даже на современном аппаратном

уровне. Тем не менее математические модели и алгоритмизация, заложенные в процедуры пре- и постобработки радарных данных, корректно могут быть перенесены на технологию параллельных вычислений. К основным процедурам обработки можно отнести, например, корегистрацию снимков [3], расчет когерентности и формирование интерферограммы [4], а также развертку фазы [5–8] и последующий расчет ЦМР [6], которые можно отнести к наиболее ресурсоемким этапам.

Оптимизации и ускорению работы, в том числе и за счет распараллеливания расчета математических алгоритмов в процедурах обработки радарных данных, посвящено большое количество научных работ [6–14].

В статьях [6–11] авторы используют технологию CUDA для улучшения производительности процедуры развертки фазы на базе различных математических моделей. В частности, рассматриваются уравнение Пуассона с весовыми коэффициентами, метод сопряженных градиентов [6, 7], метод роста регионов и отсечения ветвей [8, 9], компенсации фазового набега [10] и совмещения (корегистрации) радарных снимков [11]. В работе [12] представлен алгоритм генерации нативных (синтетических) радарных данных, симулирующих различные ошибки/шумы принимающей/передающей части космических аппаратов в зависимости от наблюдаемой земной поверхности с целью выработки программных методов коррекции.

Наряду с GPU имплементацией существуют программные реализации алгоритмов на базе параллельных вычислений на CPU. В работе [13] предлагается интеграция технологии параллельных вычислений MPI [15] с системой комплексной обработки радарных изображений Doris (Delft object-oriented radar interferometric software) [16]. Рассмотрены основные этапы процессинга InSAR/PS-InSAR технологий, включая наиболее ресурсоемкие (корегистрация, формирование интерферограммы и развертка фазы). Предложена стратегия распараллеливания декомпозиции главного/подчиненного изображений с большим количеством сегментов и частичным перекрытием границ, обеспечивающая минимальное межпроцессорное взаимодействие.

Преимуществом подходов, показанных в [6–14], является алгоритмизация решений задач пре- и постобработки, которые легко переносятся на параллельные вычисления, с помощью уже широко известных процедур, реализованных в библиотеках для GPU и CPU. Так, в работах [6, 10–15] за основу взяты элементы пакетов CuFFT, CuBLASS, CuSPARSE, PBLASS, FFTW, ScaLAPACK [17–19].

В последнее время в структуре стандартного алгоритмического аппарата DInSAR широкое распространение получил метод интерферометрии малых базовых линий (SBaS), совместно использующий длинные временные серии радарных изображений одной территории, полученные за счет повторяющейся геометрии съемки в хронологически упорядоченных промежутках времени [20]. SbaS относят к одной из наиболее ресурсоемких технологий обработки радарных данных [13]. Например, только на начальном этапе производится формирование интерферограмм на базе комплексного перемножения всех возможных пар главного/подчиненного изображений. Затем для каждой парной интерферограммы проводится развертка фазы с целью получения полного смещения в направлении на спутник. При этом чаще всего сначала применяют алгоритмы «минимальной стоимости», а затем алгоритм «роста регионов». Учитывая тот факт, что для выявления динамики и средней скорости изменения вертикальных смещений земной поверхности с погрешностью разности высот ЦМР не более чем ± 3 мм/пиксел необходимо как минимум 30 изображений, на отдельных стадиях расчетов возникает резкая деградация производительности. Экспе-

риментальные расчеты показывают время от 3 до 5 ч для 12 пар снимков небольшого разрешения в 3000×1000 пикселей.

Учитывая, что в настоящее время существует огромное количество программно реализованных высокопроизводительных алгоритмов технологических этапов обработки радарных данных, целесообразно применять их совместно в облачной инфраструктуре. Причем сама облачная инфраструктура выступает как интегратор распределенного исполнения программного кода на данных, получаемых в потоковом режиме. Технологии распределенных вычислений в облачной среде широко распространены в различных областях науки, однако нам не удалось найти применение последних в сфере тематической обработки именно радарных снимков. Поэтому задача разработки интегрального программного комплекса для пре- и постпроцессинга радарных изображений в распределенной среде с массово-параллельными механизмами (на примере Apache Spark) является, на наш взгляд, весьма актуальной для современной радарной интерферометрии.

Сравнение технологий распределенных вычислений. При выборе технологии распределенных вычислений были учтены особенности пре- и постпроцессинга радарных данных. Так, популярные технологии параллельного программирования, построенные по стандарту MPI (Message Passing Interface), например OpenMPI, не предлагают распределенного файлового хранилища, что вносит ограничение на возможные объемы обрабатываемой информации из-за необходимости применения дорогостоящих систем хранения данных и отсутствия возможности их линейного масштабирования [21].

Сейчас все бóльшую популярность приобретают облачные системы распределенной обработки данных на базе технологии MapReduce. Такие системы включают в себя распределенную файловую систему, обеспечивают отказоустойчивость при выходе из строя отдельных узлов кластера, повышают надежность хранения данных и корректность выполнения их обработки, а также возможность использования оборудования широкого назначения с низкой стоимостью.

Для сравнения были взяты различные реализации технологии MapReduce и рассмотрены применительно к задачам обработки радарных данных (табл. 1). Наиболее важным критерием при выборе вычислительной платформы являлась возможность хранения промежуточных результатов в оперативной памяти, что позволяло получить существенно бóльшую производительность в сравнении со стандартным подходом, например Hadoop MapReduce [22]. Такую функциональность предоставляет платформа Apache Spark.

Таблица 1. **Сравнение технологий распределенных вычислений**

Способ реализации	Apache Spark	Tez	Hadoop MapReduce
Интерактивный режим	Да	Да	Нет
Языки программирования	Java, Scala, Python, R	Java	Java
Работа в потоковом режиме	Да	Нет	Нет
Хранение промежуточных результатов	RAM, HDD	RAM, HDD	HDD

Важное преимущество системы Apache Spark — потоковая обработка данных, которая упрощает разработку автоматической системы пре- и постпроцессинга из-за отсутствия необходимости создавать собственную программную реализацию работы

с потоками [23]. Ее применение позволяет организовать автоматическую обработку поступающих снимков без непосредственного участия пользователя. При таком подходе процесс пре- и постобработки будет представлен в виде последовательности операций, которые необходимо выполнить для входных данных в зависимости от заданных ранее условий. Результаты этой обработки будут доступны в системе наряду с исходными данными. Таким образом, пользователю системы достаточно указать источник входных данных, а затем, по мере поступления снимков и их автоматической обработки согласно цепочке алгоритмов, на каждом этапе получать/сохранять промежуточный результат в распределенную файловую систему.

На основании результатов сравнения нами была выбрана технология массово-параллельных вычислений Apache Spark. Ее преимуществами являются:

• высокая масштабируемость, достигаемая за счет добавления новых узлов в вычислительный кластер, без необходимости внесения изменений в применяемые алгоритмы;

• встроенная возможность работы в режиме реального времени, позволяющая построить алгоритмы потоковой обработки радарных данных;

• большое количество вспомогательных программных решений, необходимых для организации системы, которая будет поддерживать полный цикл предметных задач.

Постановка задачи. Создадим распределенный программный комплекс на базе массово-параллельной технологии Apache Spark для потоковой пре- и постобработки радарных снимков со следующими функциональными особенностями:

1) возможность интеграции в программный код существующих высокопроизводительных решений для отдельных этапов (в частности, расчет когерентности, формирование интерферограмм, развертка фазы) обработки радарных ДДЗ;

2) возможность автоматического выбора высокопроизводительной параллельной технологии (например, CUDA, MPI и пр.) в расчетном ядре реализуемых алгоритмов;

3) использование общего распределенного пространства для хранения промежуточных результатов расчетов в виде бинарных данных стандартов BSQ с возможностью доступа к ним отдельных алгоритмов последующих этапов обработки с разных узлов экосистемы Apache Spark.

Программная реализация. Для решения задач пре- и постобработки радарных снимков был разработан распределенный программный комплекс на базе технологий обработки больших данных Apache BigData. Для этого были использованы языки программирования Java и Scala, методы работы с распределенными наборами данных (Spark RDD), операции над элементами которых могут выполняться параллельно. На рис. 1 представлена архитектура программного комплекса.

Разработанное решение обладает такими функциональными особенностями.

Распределенная отказоустойчивая файловая система на базе технологии HDFS позволяет хранить данные и их производные на всех этапах обработки и предоставлять доступ к ним с различных вычислительных узлов кластера. Данный подход позволил отказаться от применения дорогостоящих систем хранения и привел к увеличению доступного пространства за счет простого добавления новых узлов в кластер. Отвечает за компонент «Файловый менеджер», представленный на рис. 1. HDFS является базовым компонентом Apache Hadoop и имеет тесную интеграцию с платформой Apache Spark.

Распределенная вычислительная платформа на базе технологии Apache Spark позволяет производить параллельную обработку потоковых радарных данных

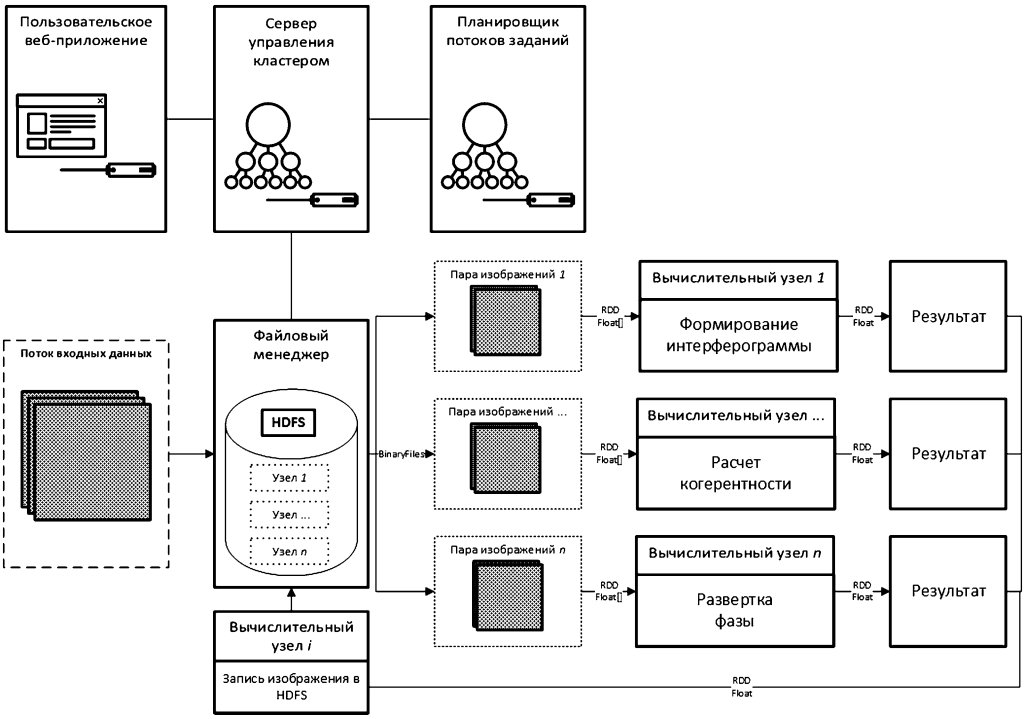

Рис. 1. Архитектура распределенного программного комплекса

на множестве узлов с максимально возможным объемом обрабатываемых данных, ограниченным только количеством узлов кластера. Масштабирование программного комплекса осуществляется за счет добавления новых узлов без изменения программной реализации используемых алгоритмов. Отвечает за компоненты «Вычислительный узел» (см. рис. 1).

Система планирования потоков работ на базе технологии Apache Oozie [24] дает возможность организовать процесс обработки потока радарных данных в виде последовательности действий, каждое из которых выполняется после успешного завершения предыдущего. Такие последовательности могут запускаться автоматически при поступлении новых данных, по запросу пользователя либо по расписанию. Отвечает за компонент «Планировщик потоков заданий» (см. рис. 1). Данная система была выбрана для организации процесса пре- и постобработки радарных данных в виде последовательности заданий, которые необходимо выполнить для входных данных в зависимости от различных условий.

Пользовательское веб-приложение на базе технологии Apache Hue [25] предоставляет возможность взаимодействия с программным комплексом. Приложение состоит из файлового менеджера, позволяющего загружать новые файлы, а также просматривать и редактировать уже существующие файлы в распределенной файловой системе. Вторым важным компонентом системы является интегрированная среда управления потоками работ с возможностью их создания и редактирования, запуска и мониторинга. Отвечает за компонент «Пользовательское веб-приложение», представленное на рис. 1. Выбор данной технологии обусловлен интеграцией со всеми приведенными выше компонентами. Она позволяет пользователю просматривать и изменять опера-

ции, выполняемые над текущими входными данными, осуществлять мониторинг состояния системы (просмотр выполняемых и выполненных работ), а также получать доступ к исходным данным и результатам их обработки через встроенный файловый менеджер, интегрированный с распределенной файловой системой HDFS.

Для организации потоковой обработки используется метод Spark Streaming API binaryRecordsStream(). С его помощью можно создать входной поток бинарных данных, который будет осуществлять мониторинг распределенной файловой системы, совместимой с HDFS на наличие новых файлов, и считывать их как бинарный файл со значением фиксированного размера байт. В качестве входных параметров используются каталог для мониторинга и размер одной записи в файле в байтах. Далее, на полученном входном потоке выполняется метод forEachRDD(). Входным параметром является функция, которая реализуется для каждого нового изображения из входного потока в виде RDD (Resilient Distributed Datasets), состоящего из значений, принимаемых конкретным пикселом снимка.

Для загрузки исходных данных из файловой системы HDFS запускается метод Spark API binaryFiles(). Входным параметром служит путь к каталогу с исходными данными (пакет). Результатом работы является RDD, состоящий из пар, каждая их которых содержит имя файла и соответствующий ему поток данных. С помощью метода keyBy() для каждого элемента пары определяется его ключ, как имя файла без префикса. Получаемый таким образом RDD, содержащий бинарную последовательность интерферограммы и значений ее когерентности, имеет уникальный ключ. Текущие значения RDD группируются по ключу методом API groupByKey(). Передача бинарной последовательности и запуск программного кода алгоритма развертки фазы производятся с помощью функции map(). Программный код функции map() проверяет аппаратную доступность технологии CUDA на каждом узле, в зависимости от этого выбирается имплементация алгоритма с использованием CPU или GPU. Разработанному алгоритму передаются два массива байт (интерферограмма и значения ее когерентности), содержащиеся в RDD, полученный результат сохраняется в распределенную файловую систему HDFS напрямую.

В зависимости от реализации конкретного алгоритма деление входных данных осуществляется такими способами:

1) отдельные файлы радарных данных, предназначенные для обработки, делятся на области; результат для каждой области вычисляется на отдельном узле, полученные данные объединяются и сохраняются в готовое изображение;

2) отдельное изображение обрабатывается целиком на вычислительном узле без деления на области; параллелизация достигается за счет обработки большого количества изображений на множестве узлов.

Рассмотрим реализацию представленной выше обобщенной диаграммы потоков данных на примере алгоритма расчета фазы [26]. Блок-схему предложенного алгоритма иллюстрирует рис. 2.

Как показано выше на диаграмме, в процессе работы алгоритма проводится параллельная развертка интерферометрической фазы, а количество одновременно выполняемых расчетов определяется количеством узлов в кластере. Входными данными являются массив значений интерферограммы в каждой точке изображения и массив величин когерентности каждого значения интерферограммы. В общем виде этап развертки фазы состоит из следующих шагов:

1. Выбираются начальные точки с наибольшей когерентностью, они объявляются развернутыми и образуют регионы. Региону с наибольшей когерентностью начальной

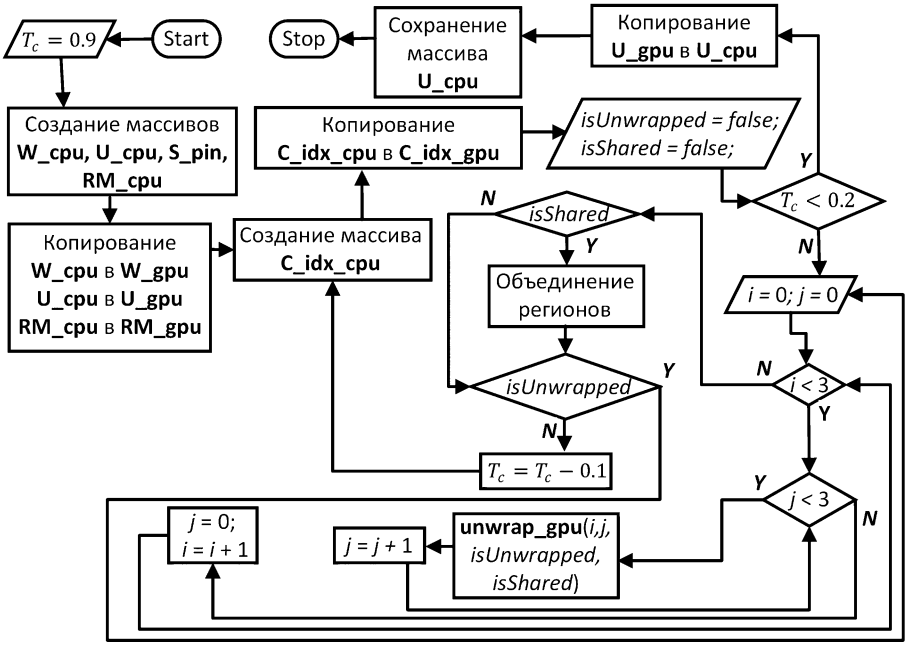

Рис. 2. Алгоритм развертки интерферометрической фазы

точки присваивается номер 1 и так далее, при уменьшении значения когерентности начальной точки увеличивается номер региона.

2. Для каждой развернутой точки ищутся соседние неразвернутые (целевые) точки, образующие кольца роста регионов. Назовем этот процесс итерацией роста.

3. Для каждой целевой точки кольца роста вычисляется предсказываемое значение по формуле

$$\phi^\rho = \frac{\sum_{\kappa=1}^{N_\upsilon} \omega^\kappa \phi_\kappa^\rho}{\sum_{\kappa=1}^{N_\upsilon} \omega^\kappa},$$

где $N_u > 1$ (за исключением области начальных точек), N_u — количество соседних пикселей с развернутой фазой; $\phi_k^p = 2\phi[k] - \phi[k']$, $\omega^k = 1.0$, если на пути предсказания к точке лежат два развернутых пиксела; $\phi_k^p = \phi[k]$, $\omega^k = 0.5$ — если один развернутый пиксел; $\phi[k]$ и $\phi[k']$ — значения развернутых фаз на пути предсказания соответственно.

4. Рассчитывается число неоднозначности m

$$m = \mathrm{nint}((\phi^p - \varphi_\omega)/2\pi),$$

в которой φ_ω — значение свернутой (относительной) фазы целевого пиксела, nint — оператор взятия ближайшего целого.

5. Рассчитывается величина развернутой (абсолютной) фазы $\phi_u = \varphi_\omega + 2$.

6. Значение развернутой фазы в целевой точке принимается, если выполняются условия надежности $d^p < T_p$, $d_u < T_u$ и когерентность в целевой точке больше T_c, где $d^p = \frac{\sum_{k=1}^{N_u} \omega^k |\phi_k^p - \phi^p|}{\sum_{k=1}^{N_u} \omega^k}$, $d_u = |\phi_u - \phi^p|$, $T_p = T_u = \pi/2$, $T_p = T_u = \pi/2$, $T_c = 0.9 \ldots 0.2$, T_c — параметр релаксации. Если два региона имеют точки пересечения, запускается процедура их объединения.

7. Определяется количество общих точек ($N_o v$).

8. Для каждой общей точки вычисляется разность их чисел неоднозначности ($D_m d = m_R1 - m_R2$), m_R1, m_R2 — числа неоднозначности общей точки в регионах с номерами R1 и R2 соответственно.

9. Находятся мода значений разностей $D_m d$ и количество точек, образующих ее (N_c).

10. Если $\frac{N_c}{N_{ov}} \geqslant T_r r$ и $N_o v \geqslant T_r n$, где $T_r r = 0.75$, $T_r n = 3$, то регионы объединяются. В точках склейки величина абсолютной фазы берется для региона с наименьшим номером, остальные точки исключаются из алгоритма. Если условия не выполнены, исключаются все общие точки.

В табл. 2 дано описание объектов (массивов), используемых в программной реализации усовершенствованного алгоритма роста регионов.

Таблица 2. **Основные объекты (массивы) алгоритма**

Название объекта, массива	Описание
isUnwrapped	Флаг остановки процедуры итераций роста
isShared	Флаг запуска процедуры объединения регионов
Nu	Счетчик количества развернутых соседних точек
D=−N−1,−N,−N+1,−1,1,N−1, N,N+1 и D1=−2N−2,−2N, −2N+2,−2,2,2N−2,2N,2N+2	Массивы констант для получения значений индексов соседних точек к целевой
sum(arg, index) и round_int(arg)	Функции нахождения суммы аргументов с указанными индексами и округление до ближайшего целого соответственно
C_idx_gpu	Массив индексов целевых точек W_gpu, удовлетворяющих текущим параметрам надежности (когерентность больше T_c)
U_gpu	Массив значений развернутых фаз
RM_gpu	Массив номеров регионов
S_pin	Массив общих точек

В табл. 3 приведены реализованные алгоритмы обработки радарных данных с указанием особенностей их распараллеливания.

Таблица 3. **Особенности реализации различных алгоритмов**

Алгоритм	Входные данные	Особенности реализации
Расчет фазы	Бинарные представления радарных данных аппарата Cosmo-SkyMed уровня L1A в формате (.hdr + SLCBinary) на примере Exelis ENVI	Изображение делится на отдельные значения (точки) последовательно, каждое изображение рассчитывается на множестве узлов по частям
Формирование интерферограммы		Изображение делится на регионы заданного размера (7×7 точек), каждое изображение рассчитывается на множестве узлов по частям
Расчет когерентности		
Развертка фазы		Изображение обрабатывается целиком на отдельном вычислительном узле, производится параллельная обработка множества изображений

Расчетная часть представленных алгоритмов реализована в соответствии с аналогичными схемами в основных системах обработки радарных данных. Так, расчет фазы и формирование интерферограммы производятся согласно выражениям

$$Phase_{i,j} = \arctan\left(\frac{(float)I_{i,j}}{(float)(I_{i,j} \gg 32)}\right),$$

$$I_{i,j} = (float)(SLC^M_{i,j} * SLC^S_{i,j}) + (float)(SLC^M_{i,j} \gg 8) * (SLC^S_{i,j} \gg 8),$$

$$I_{i,j} \ll 32 = (float)(-(SLC^M_{i,j} * (SLC^S_{i,j} \gg 8)) + (float)((SLC^M_{i,j} \gg 8) * SLC^S_{i,j})),$$

$$I_{i,j} = (I_{i,j})|((I_{i,j}) \ll 32),$$

а расчет когерентности — по формуле

$$C_{i,j} = \frac{\frac{1}{49}\sum_{i=0}^{7}\sum_{j=0}^{7}|I_{i,j}|}{(\frac{1}{49}\sum_{i=0}^{7}\sum_{j=0}^{7}|SLC^M_{i,j}|^2) * (1/49\sum_{i=0}^{7}\sum_{j=0}^{7}|SLC^s_{i,j}|^2)}.$$

Для развертки интерферометрической фазы применяется алгоритм роста регионов. Особенностью этого способа обработки радарных данных является использование распределенных технологий (HDFS и Apache Spark), что позволяет применять алгоритмы для потоков данных. На рис. 3 показан графический интерфейс разработанного алгоритма в виде workflow-задания развертки интерферометрической фазы.

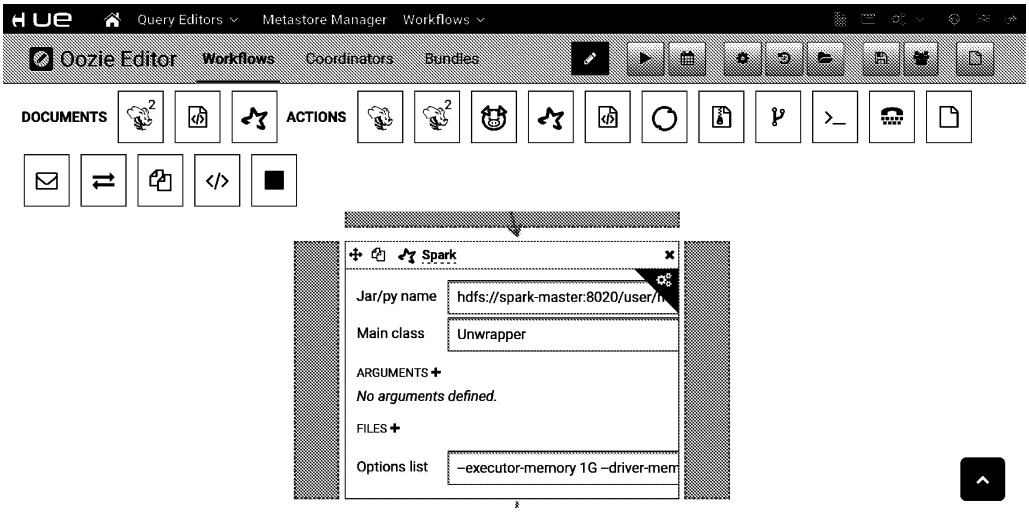

Рис. 3. Графический интерфейс workflow-задания развертки интерферометрической фазы

Результаты тестирования. Разработанные алгоритмы были протестированы на кластере из четырех и восьми узлов, созданном с использованием CDH5-дистрибутива среды Hadoop. Также вычисления были произведены на отдельном узле без применения распределенного подхода.

При создании кластера была выбрана версия среды CDH 5.7.0, которая включает следующие компоненты: Apache HDFS 2.6, Apache Spark 1.6.0, Apache Hue 3.9, Apache Oozie 1.7. Компоненты хранения и обработки данных были установлены на каждом узле кластера (slave-узлы), в то время как система управления потоками работ и пользовательское веб-приложение были установлены только на одном узле (master-узел). Все узлы кластера были подключены к локальной вычислительной сети с пропускной способностью 1 ГБит/с. При проведении расчетов на отдельном компьютере был

использован только вычислительный компонент представленного программного комплекса (Apache Spark), так как остальные компоненты целесообразно применять только для кластера.

В качестве тестовых данных были выбраны радарные изображения размера 16 384 × 16 384 точек, с объемом одного файла 2 Гб. Для расчета относительной фазы было взято одно изображение, а для формирования интерферограммы, расчета когерентности и развертки фазы — пара изображений, что обусловлено особенностями работы таких алгоритмов.

Результаты тестирования приведены на рис. 4. Каждый узел кластера соответствует следующей конфигурации: Ubuntu 16.04; Oracle JDK 1.8; Intel Xeon E5-2620 v2 @ 2.10GHz; 6GB RAM.

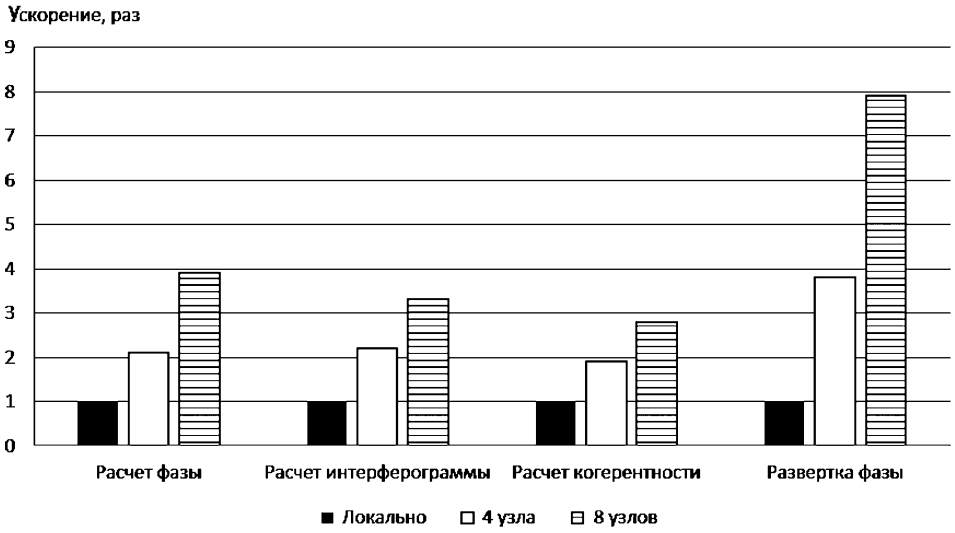

Рис. 4. Результаты тестирования

Важно отметить, что для алгоритмов расчета фазы, формирования интерферограммы и определения когерентности применяется разделение на уровне изображения, за счет чего и достигается ускорение при обработке отдельного снимка относительно локального подхода. Алгоритм развертки фазы предполагает проведение всех необходимых расчетов для одной пары изображений на одном вычислительном узле, а ускорение достигается за счет возможности одновременной обработки изображений, количество которых равно количеству узлов кластера. Так, при количестве узлов кластера, равном 8, в среднем возможно достижение 8-кратного прироста скорости работы для восьми пар изображений по сравнению с их последовательной обработкой на отдельном вычислительном узле.

Результаты тестирования показывают возможность повышения производительности представленных алгоритмов при увеличении количества узлов кластера без внесения изменений в их реализацию, что оправдывает применение распределенного подхода для решения данной задачи.

Заключение. В результате анализа различных подходов, применяемых при обработке радарных данных, а также обзора технологий распределенных вычислений был предложен и реализован распределенный программный комплекс на базе массово-параллельной технологии Apache Spark для потоковой пре- и постобработки

снимков. По сравнению с традиционными подходами к обработке радарных изображений, в которых параллельные вычисления либо не используются, либо применяются только для повышения производительности расчетов, предложенное решение ориентировано на обработку большого количества потоковых данных. Программная реализация комплекса содержит веб-интерфейс, позволяющий пользователю взаимодействовать с кластером, получая доступ к распределенной файловой системе, а также создавать и исполнять существующие потоки работ, существенно уменьшая вычислительные затраты и увеличивая эргономику работы с системой.

Литература

1. *Елизаветин И. В., Шувалов Р. И., Буш В. А.* Принципы и методы радиолокационной съемки для целей формирования цифровой модели местности // Геодезия и картография. 2009. № 1. С. 39–45.

2. *Ferretti A., Monti-Guarnieri A., Prati C.* et al. InSAR Principles: Guidelines for SAR interferometry processing and interpretation // URL: http://www.esa.int/esapub/tm/tm19/TM-19_ptA.pdf (дата обращения: 02.08.2016).

3. *Zhengxiao Li, Bethel J.* Image coregistration in SAR interferometry // The Internati-onal Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Beijing, 2008. Vol. XXXVII. Pt B1. P. 433–438.

4. *Massonnet D., Feigl K. L.* Radar interferometry and its application to changes in the earth's surface // Reviews of Geophysics. 1998. Vol. 36, Iss. 4. P. 441–500.

5. *Costantini M., Farina A., Zirilli F.* A fast phase unwrapping algorithm for SAR interferometry // IEEE Trans. GARS. 1999. Vol. 37, N 1. P. 452–460.

6. *Mistry P., Braganza S., Kaeli D., Leeser M.* Accelerating phase unwrapping and affine transformations for optical quadrature microscopy using CUDA // Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units. GPGPU: Conference. Washington, D.C., USA: ACM, 2009. P. 28–37.

7. *Karasev P. A., Campbell D. P., Richards M. A.* Obtaining a 35x Speedup in 2D phase unwrapping using commodity graphics processors // Radar Conference. IEEE. 2007. P. 574–578.

8. *Wu Z., Ma W., Long G., Li Y., Tang Q., Wang Z.* High performance two-dimensional phase unwrapping on GPUs // Proceedings of the 11th ACM Conference on Computing Frontiers — CF'14. New York, NY, USA: ACM, 2014. P. 35:1–35:10.

9. *Xin-Liang S., Xiao-Chun X.* GPU acceleration of range alignment based on minimum entropy criterion // Radar Conference. IET International. 14–16 April 2013. P. 1–4.

10. *Guerriero A., Anelli V. W., Pagliara A., Nutricato R., Nitti D. O.* High performance GPU implementation of InSAR time-consuming algorithm kernels // Proceedings of the 1st WORKSHOP on the State of the art and Challenges of Research Efforts at POLIBA. Bari, Italy: Politecnico di Bari, 2014. P. 383.

11. *Zhang F., Wang B., Xiang M.* Accelerating InSAR raw data simulation on GPU using CUDA // Geoscience and Remote Sensing Symposium (IGARSS). IEEE International. Bari, Italy: Politecnico di Bari, 25–30 July 2010. P. 2932–2935.

12. *Marinkovic P. S., Hanssen R. F., Kampes B. M.* Utilization of parallelization algorithms in InSAR/PS-InSAR processing // Proceedings of the 2004 Envisat ERS Symposium (ESA SP-572). Salzburg, Austria: ESA, 6–10 September 2004. P. 1–7.

13. *Sheng G., Qi-Ming Z., Jian J., Cun-Ren L., Qing-xi T.* Parallel processing of InSAR interferogram filtering with CUDA programming // Zhongguo Cehui Kexue Yanjiuyan, China. 2015. Vol. 40, N 1. P. 67–88.

14. *Верба В. С., Неронский Л. Б., Осипов И. Г., Турук В. Э.* Радиолокационные системы землеобзора космического базирования. М.: Радиотехника, 2010. 675 с.

15. *Gabriel E., Fagg G. E., Bosilca G.* et al. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation // URL: https://www.open-mpi.org/papers/euro-pvmmpi-2004-overview/euro-pvmmpi-2004-overview.pdf (дата обращения: 30.06.2016).

16. *Kampes B., Hanssen R., Perski Z.* Radar Interferometry with Public Domain Tools presentation // URL: http://doris.tudelft.nl/Literature/kampes_fringe03.pdf (дата обращения: 30.06.2016).

17. *Frigo M., Johnson S. G.* FFTW: An Adaptive Software Architecture for the FFT // ICASSP conference proceedings. Seattle, Washington, USA: IEEE, 15 May 1998. Vol. 3. P. 1381–1384.

18. *Larkin J.* Fast GPU Development with CUDA Libraries // URL: https://www.olcf.ornl.gov/wp-content/uploads/2013/02/GPU_libraries-JL.pdf (дата обращения: 30.06.2016).

19. *Demmel J., Dongarra J.* ST-HEC: Reliable and scalable software for linear algebra computations on High End Computers // URL: https://people.eecs.berkeley.edu/ demmel/Sca-LAPACK-Proposal.pdf (дата обращения: 30.06.2016).

20. *Феоктистов А. А., Захаров А. И., Гусев М. А., Денисов П. В.* Исследование возможностей метода малых базовых линий на примере модуля SBaS программного пакета SARScape и данных PCA ASAR/ENVISat и PALSAR/ALOS. Ч. 1. Ключевые моменты метода // Журн. радиоэлектроники. 2015. № 9. С. 1–26.

21. *Reyes-Ortiz J. L., Oneto L., Anguita D.* Big Data analytics in the cloud: Spark on Hadoop vs MPI/OpenMP on Beowulf // INNS Conference on Big Data 2015 Program. San Francisco, USA. 8–10 August 2015. P. 121–130.

22. *Kannan P.* Beyond Hadoop MapReduce Apache Tez and Apache Spark // URL: http://www.sjsu.edu/people/robert.chun/courses/CS259Fall2013/s3/F.pdf (дата обращения: 02.08.2016).

23. *Nathan P.* Real-Time analytics with Spark Streaming // URL: http://viva-lab.ece.virginia.edu/foswiki/pub/InSAR/RitaEducation/InSAR Technology Literature Search.pdf (дата обращения: 02.08.2016).

24. *Nagler E.* Introduction to Oozie // Apache Oozie Documentation. URL: http://www.cse.buffalo.edu/bina/cse487/fall2011/Oozie.pdf (дата обращения: 02.08.2016).

25. *Jhajj R.* Apache Hadoop Hue Tutorial // URL: https://examples.javacodegeeks.com/enterprise-java/apache-hadoop/apache-hadoop-hue-tutorial/ (дата обращения: 02.08.2016).

26. *Потапов В. П., Попов С. Е.* Высокопроизводительный алгоритм роста регионов для развертки интерферометрической фазы на базе технологии CUDA // Программная инженерия. 2016. № 2. С. 61–74.

Для цитирования: *Потапов В. П., Костылев М. А., Попов С. Е.* Потоковая обработка радарных данных в распределенной среде Apache Spark // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. 2017. Т. 13. Вып. 2. С. 168–181. DOI: 10.21638/11701/ spbu10.2017.204

References

1. Elizavetin I. V., Shuvalov R. I., Bush V. A. Principy i metody radiolokacionnoj s'emki dlja celej formirovanija cifrovoj modeli mestnosti [Principles and methods of SAR Interferometry for the purpose of forming a digital elevation model]. *Geodesy and cartography*, 2009, no. 1, pp. 39–45. (In Russian)

2. Ferretti A., Monti-Guarnieri A., Prati C. et al. *InSAR Principles: Guidelines for SAR Interferometry Processing and Interpretation.* Available at: http://www.esa.int/esapub/tm/tm19/TM-19_ptA.pdf (accessed: 02.08.2016).

3. Zhengxiao Li, Bethel J. Image coregistration in SAR interferometry. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences.* Beijing, 2008, vol. XXXVII, pt B1, pp. 433–438.

4. Massonnet D., Feigl K. L. Radar interferometry and its application to changes in the earth's surface. *Reviews of Geophysics*, 1998, vol. 36, issue 4, pp. 441–500.

5. Costantini M., Farina A., Zirilli F. A fast phase unwrapping algorithm for SAR interferometry. *IEEE Trans. GARS*, 1999, vol. 37, no. 1, pp. 452–460.

6. Mistry P., Braganza S., Kaeli D., Leeser M. Accelerating phase unwrapping and affine transformations for optical quadrature microscopy using CUDA. *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units. GPGPU. Conference.* Washington, D.C., USA, ACM, 2009, pp. 28–37.

7. Karasev P. A., Campbell D. P., Richards M. A. Obtaining a 35x speedup in 2d phase unwrapping using commodity graphics processors. *Radar Conference.* IEEE, 2007, pp. 574–578.

8. Wu Z., Ma W., Long G., Li Y., Tang Q., Wang Z. High performance two-dimensional phase unwrapping on GPUs. *Proceedings of the 11th ACM Conference on Computing Frontiers — CF'14.* New York, NY, USA, ACM, 2014, pp. 35:1–35:10.

9. Xin-Liang S., Xiao-Chun X. GPU acceleration of range alignment based on minimum entropy criterion. *Radar Conference. IET International*, 14–16 April 2013, pp. 1–4.

10. Guerriero A., Anelli V. W., Pagliara A., Nutricato R., Nitti D. O. High performance GPU implementation of InSAR time-consuming algorithm kernels. *Proceedings of the 1st WORKSHOP on the State of the art and Challenges of Research Efforts at POLIBA.* Bari, Italy, Politecnico di Bari, 2014, pp. 383.

11. Zhang F., Wang B., Xiang M. Accelerating InSAR raw data simulation on GPU using CUDA. *Geoscience and Remote Sensing Symposium* (*IGARSS*). *IEEE International.* Bari, Italy, Politecnico di Bari, 25–30 July 2010, pp. 2932–2935.

12. Marinkovic P. S., Hanssen R. F., Kampes B. M. Utilization of parallelization algorithms in InSAR/PS-InSAR processing. *Proceedings of the 2004 Envisat ERS Symposium* (*ESA SP-572*). Salzburg, Austria, ESA, 6–10 September 2004, pp. 1–7.

13. Sheng G., Qi-Ming Z., Jian J., Cun-Ren L. Qing-xi T. Parallel processing of InSAR interferogram filtering with CUDA programming. *Zhongguo Cehui Kexue Yanjiuyan,* China, vol. 40, no. 1, pp. 67–88.

14. Verba V. S., Neronskij L. B., Osipov I. G., Turuk V. Je. *Radiolokacionnye sistemy zemleobzora kosmicheskogo bazirovaniya* [*Space-based radio location systems of Earth Observation*]. Moscow, Radiotechnica Publ., 2010, 675 p. (In Russian)

15. Gabriel E., Fagg G. E., Bosilca G. et al. *Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation.* Available at: https://www.open-mpi.org/papers/euro-pvmmpi-2004-overview/euro-pvmmpi-2004-overview.pdf (accessed: 30.06.2016).

16. Kampes B., Hanssen R., Perski Z. *Radar Interferometry with Public Domain Tools presentation.* Available at: http://doris.tudelft.nl/Literature/kampes_fringe03.pdf (accessed: 30.06.2016).

17. Frigo M., Johnson S. G., FFTW: An Adaptive software architecture for the FFT. *ICASSP conference proceedings.* Seattle, Washington, USA, IEEE, 15 May 1998, vol. 3, pp. 1381–1384.

18. Larkin J. *Fast GPU Development with CUDA Libraries.* Available at: https://www.olcf.ornl.gov/wp-content/uploads/2013/02/GPU_libraries-JL.pdf (accessed: 30.06.2016).

19. Demmel J., Dongarra J. *ST-HEC*: *Reliable and Scalable software for linear algebra computations on High End Computers.* Available at: https://people.eecs.berkeley.edu/ demmel/Sca-LAPACK-Proposal.pdf (accessed: 30.06.2016).

20. Feoktistov A. A., Zaharov A. I., Gusev M. A., Denisov P. V. Issledovanie vozmozhnostej metoda malyx bazovyx linij na primere modulya SBaS programmnogo paketa SARScape i dannyx RSA ASAR/ENVISat i PALSAR/ALOS. Ch. 1. Klyuchevye momenty metoda [Investigation of the possibilities of the method of small baselines technique on the example of the SBaS module of the software package SARScape and the data of the RSA ASAR/ENVISat and PALSAR/ALOS. Pt 1. Key points of the method]. *Journal of Radioelectronics*, 2015, no. 9, pp. 1–26. (In Russian)

21. Reyes-Ortiz J. L., Oneto L., Anguita D. Big Data analytics in the cloud: Spark on Hadoop vs MPI/OpenMP on Beowulf. *INNS Conference on Big Data* Program. San Francisco, USA, 8–10 August 2015, pp. 121–130.

22. Kannan P. *Beyond Hadoop MapReduce Apache Tez and Apache Spark.* Available at: http://www.sjsu.edu/people/robert.chun/courses/CS259Fall2013/s3/F.pdf (accessed: 02.08.2016).

23. Nathan P. *Real-Time Analytics with Spark Streaming.* Available at: http://viva-lab.ece.virginia.edu/foswiki/pub/InSAR/RitaEducation/InSAR Technology Literature Search.pdf (accessed: 02.08.2016).

24. Nagler E. *Introduction to Oozie. Apache Oozie Documentation.* Available at: http://www.cse.buffalo.edu/bina/cse487/fall2011/Oozie.pdf (accessed: 02.08.2016).

25. Jhajj R. *Apache Hadoop Hue Tutorial.* Available at: https://examples.javacodegeeks.com/enterprise-java/apache-hadoop/apache-hadoop-hue-tutorial/ (accessed: 02.08.2016).

26. Potapov V. P., Popov S. E. Vysokoproizvoditel'nyj algoritm rosta regionov dlya razvertki interferometricheskoj fazy na baze texnologii CUDA [High-Performance Region-Growing Algorithm for InSAR Phase Unwrapping Based on CUDA]. *Software engineering,* 2016, no. 2, pp. 61–74. (In Russian).

For citation: Potapov V. P., Kostylev M. A., Popov S. E. The streaming processing of sar data in distributed environment with Apache Spark. *Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes*, 2017, vol. 13, iss. 2, pp. 168–181. DOI: 10.21638/11701/spbu10.2017.204

Статья рекомендована к печати проф. В. Ю. Добрыниным.

Статья поступила в редакцию 15 сентября 2016 г.

Статья принята к печати 11 апреля 2017 г.